UNIVERSIDAD RAFAEL LANDÍVAR
FACULTAD DE INGENIERÍA
INGENIERÍA EN INFORMÁTICA Y SISTEMAS
LENGUAJES FORMALES Y AUTÓMATAS
CATEDRÁTICO: ING. MOISÉS ALONSO

FASE 3:
GENERADOR DE SCANNER

JOSÉ FERNANDO OLIVA MORALES
CARNÉ:1251518

GUATEMALA, 6 DE MAYO DE 2020

INTRODUCCIÓN

Como parte de la tercera fase el proyecto del curso LENGUAJES FORMALES Y AUTÓMATAS, la cual corresponde a un generador de scanner que permita generar un programa a partir del archivo de la gramática que haya sido analizada en el proyecto, y que, a partir de la tabla de estados se genere el autómata que permita la generación del código para analizar dicha gramática y determine si una cadena es válida o no.

Para ello, se implementa un algoritmo que permite la creación genérica de programas que según la gramática analizada determine si una cadena es válida o no bajo dichas reglas. Además, el programa generado muestra en pantalla a que token o action pertenece cada elemento de la cadena de entrada.

OBJETIVOS

General

- Implementar un algoritmo que permita la generación de programas a partir de un archivo que contiene una gramática.

Específicos

- Elaborar de manera genérica cada una de las validaciones para cada estado.
- Implementar un algoritmo que permite determinar a que token o action de la gramática pertenece cada elemento de la cadena analizada.
- Determinar si una cadena es valida o no bajo el programa generado según las reglas de la gramática analizada.

ALGOTIMO: GENERAR PROGRAMA A PARTIR DE AUTOMATA FINITO DETERMINISTA

Por: Fernando Oliva (feroliv4z@gmail.com).

ENTRADAS

- Autómata finito determinista generado previamente.

SALIDAS

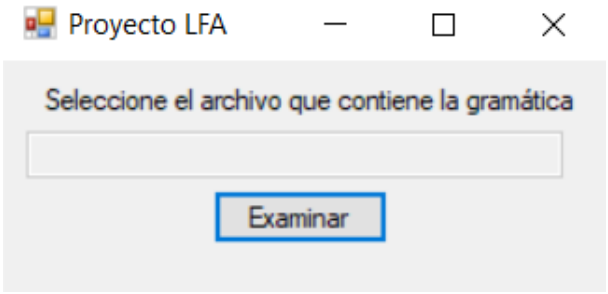
- Archivo .cs generado a partir de la tabla de transiciones del autómata finito determinista.

PROCESO

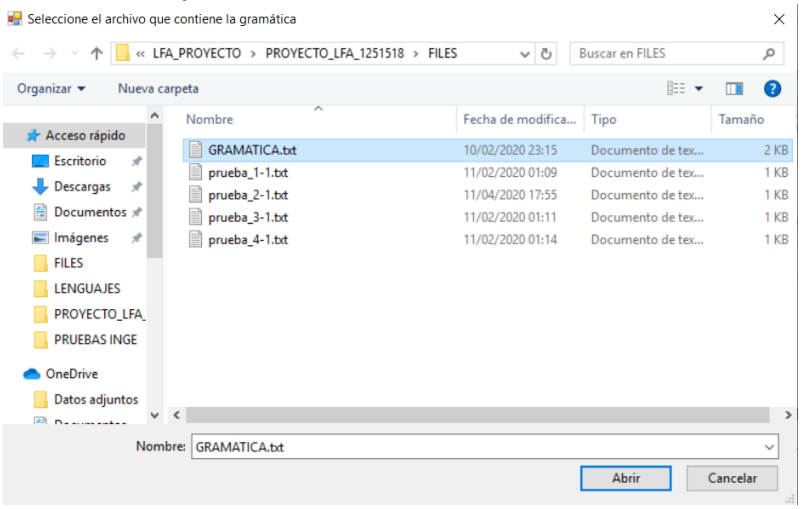
1. String content
2. Para cada estado en automata
   a. string auxContent += "boll acceptation =" + estado.acceptation + ";\r\naceptado=acceptation;\r\nvar transitions = new List<Transition>();\r\n"
   b. para cada transicion en estado
      i. aux.Content += "transitions.Add(new Transition(" + transicion.simbolo + "," + transicion.status + "));\r\n"
   c. auxContent += "foreach (var item in transitions)\r\n{\r\nif (item.Simbol.Replace(" + Convert.ToChar(34) + "'" + Convert.ToChar(34) + "," + Convert.ToChar(34) + Convert.ToChar(34) + ") == token){"
   d. auxContent += "estado = item.Status;\r\ntkn.Add(token);\r\naceptado = acceptation;\r\nenc = true;\r\nbreak;\r\n}\r\n}\r\nif (enc)\r\nbreak;\r\nelse\r\n{\r\nforeach (var item in transitions)\r\n{\r\n"
   e. auxContent += "if (item.Simbol.Replace(" + Convert.ToChar(34) + "'" + Convert.ToChar(34) + "," + Convert.ToChar(34) + Convert.ToChar(34) + ") == buscarEnSets(token)){"
   f. auxContent += "estado = item.Status;\r\ntkn.Add(buscarEnSets(token));\r\naceptado = acceptation;\r\nbreak;\r\n}\r\n else if (buscarEnSets(token) == " + Convert.ToChar(34) + Convert.ToChar(34) + ")\r\n {\r\naceptado = false;\r\nbreak;\r\n}\r\nelse { aceptado = false; }\r\n}\r\n"
   g. content += "case " + estado.numero + ": { "+auxContent+"\r\n} break; \r\n}\r\n"
3. Escribir content en archivo

MANUAL DE USUARIO

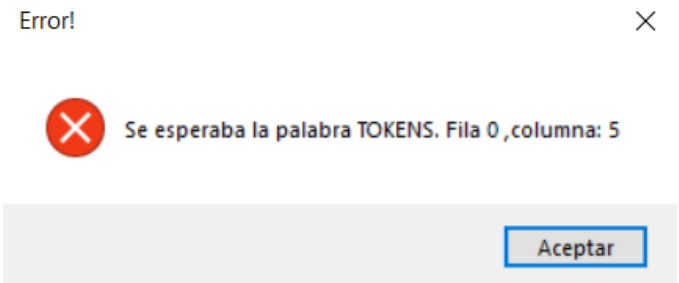
1. Al inicializar el programa, se muestra la siguiente ventana, en la cual se indica que debe seleccionarse el archivo que contenga la gramática que se desee analizar. Debe presionarse el botón “Examinar” para proceder a seleccionar el archivo.



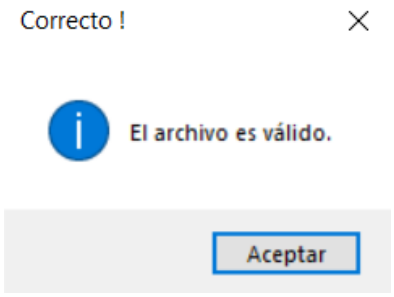
2. Al presionar el botón “Examinar”, se muestra un explorador de archivos en el que se debe seleccionar el archivo deseado. Para ello se requiere presionar el botón “Abrir en el explorador”



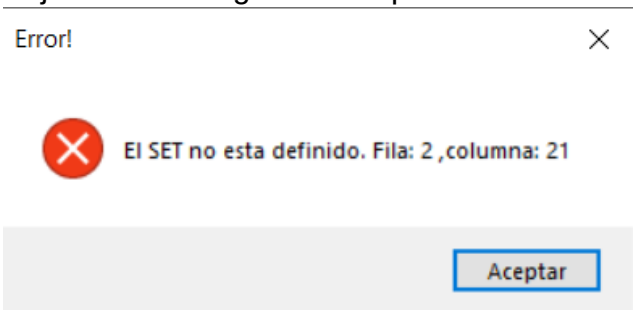
3. Primero, el contenido del archivo seleccionado pasa por el analizador léxico, en el que en el caso de encontrar un error se muestra un mensaje similar al siguiente:



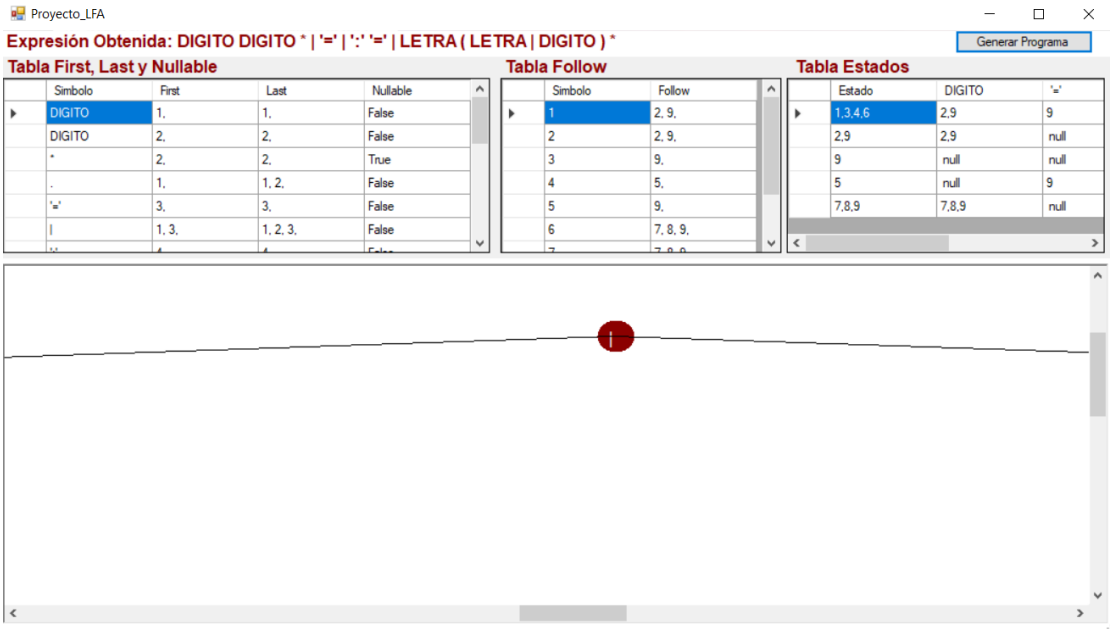
4. Si el analizador léxico da a conocer que el archivo es correcto, se muestra el siguiente mensaje en pantalla.



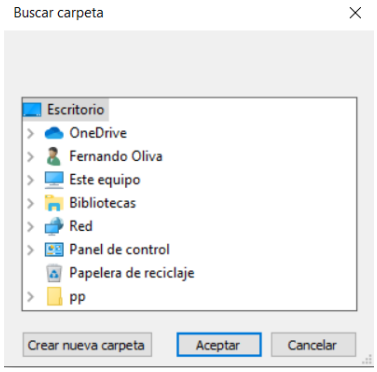
5. Por su parte, si el analizador sintáctico encuentra un error en el archivo, se muestra un mensaje similar al siguiente en pantalla.



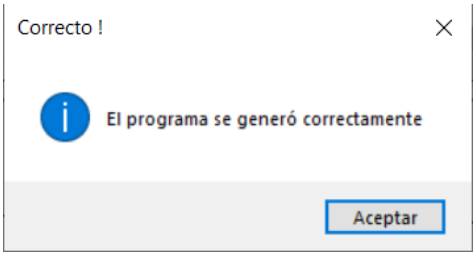
6. Si el analizador sintáctico, da a conocer que el archivo cumple con las reglas correspondientes, se muestra el resultado en pantalla. El resultado incluye, la expresión regular obtenida de la sección tokens del archivo, la tabla de first, last y follow, y la tabla de estos correspondiente. Adicionalmente, se muestra una representación gráfica del árbol de la expresión regular obtenida, para ser visualizado debe desplazarse sobre dicha sección.



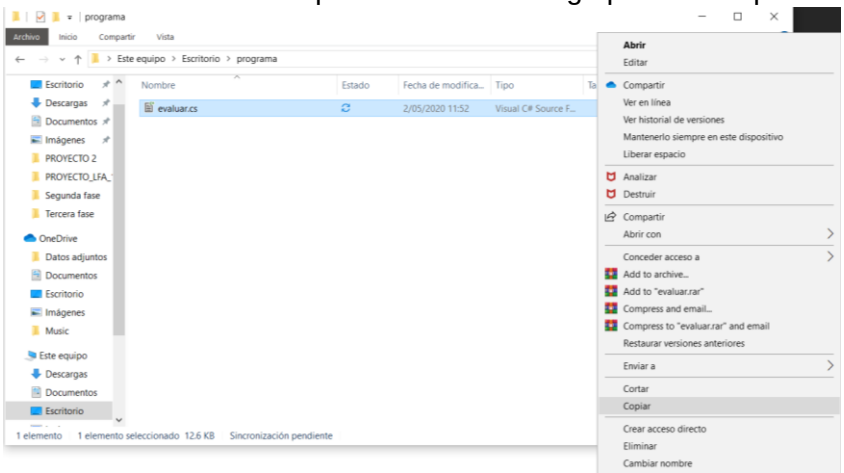
7. Seguidamente, se debe presionar el botón “Generar programa” con el fin de que se cree un archivo que contenga el código que permita analizar si una cadena es válida o no según la gramática que se ha cargado inicialmente. Al presionar el botón se abrirá la siguiente ventana, en la que se debe seleccionar el fichero en el que se desea guardar el archivo con el código generado.



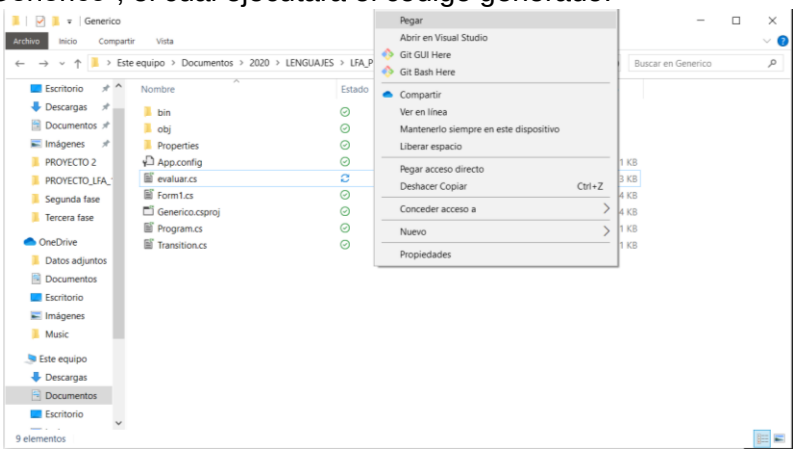
8. Al seleccionar el fichero donde guardar el archivo y presionar “aceptar”, de muestra el siguiente mensaje.



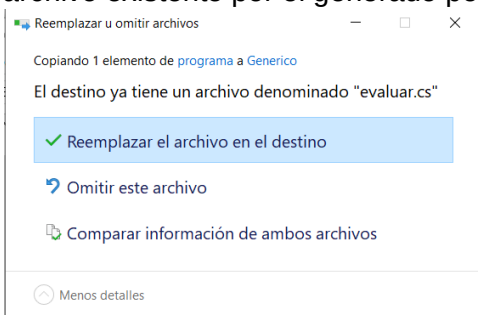
9. Luego, debe ubicarse el archivo que contiene el código para ser copiado.



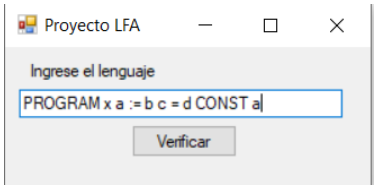
10. Luego de copiar el archivo, este debe ser pegado en la carpeta que contiene el proyecto “Genérico”, el cual ejecutará el código generado.



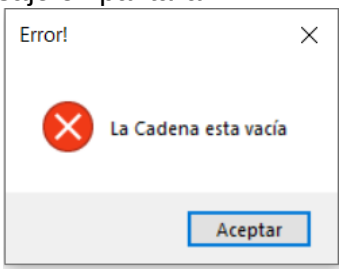
11. Se debe reemplazar el archivo existente por el generado por el programa.



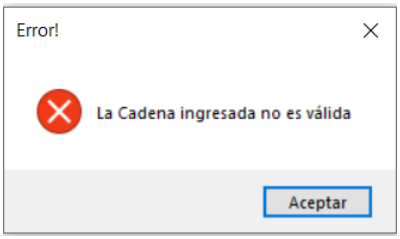
12. Al ejecutar el proyecto “Genérico”, se mostrará en pantalla la siguiente ventana en la que se debe ingresar la cadena que se desee analizar. Cada elemento debe estar a continuación por un espacio. Para analizar la cadena ingresada se debe presionar el botón “Verificar”.



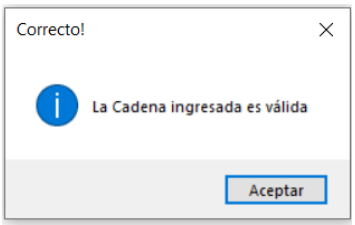
13. Si se presiona el botón sin previamente introducir una cadena en el cuadro de texto, se muestra el siguiente mensaje en pantalla.



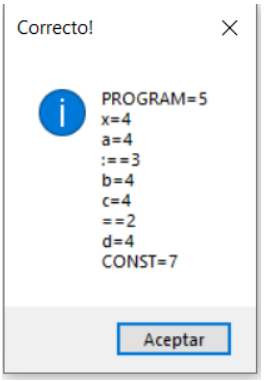
14. De lo contario, la cadena pasa a ser analizada. Si el resultado indicado por el autómata es que la cadena no cumple con la gramática generada, se muestra el siguiente mensaje en pantalla.



15. Por otro lado, si la cadena analizada cumple con la gramática generada, se muestra el siguiente mensaje.



16. Al presionar el botón aceptar, se muestra en pantalla el listado de los tokens ingresados en la cadena y a que identificador de la sección “TOKENS” o “ACTIONS” del archivo de la gramática pertenecen.



BIBLIOGRAFÍA

Harold Abelson, Gerald Jay Sussman, and Julie Sussman. Structure and Interpretation of Computer Programs. MIT Press, 1996.

Niklaus Wirth. The design of a Pascal compiler. Software - Practice and Experience, 1971.

Torben Ægidius Mogensen, Basics Of Compiler Design, University of Copenhagen, Denmark, 2000.